Approximating Square Roots

Dallas Foster

University of Utah

u0809485

December 2014

The history of approximating square roots is one of the oldest examples of numerical approximations found. The Babylonian method, a very early and good approximation, has been around since ancient times. This paper will test and analyze a new method for approximating square roots that is based off of the inequalities of the arithmetic, geometric, and harmonic means. First, the derivation of the method will be shown and the proof of the convergence of this method will be demonstrated. Following, an in-depth analysis of the method will be conducted: including a discussion of the rate of convergence, and the effect of initial guesses. The new method will then be tested against the Babylonian method in a variety of manners. Finally, an extension of this method will be given that can calculate the nth root of a number.

I. Method Derivation

With a brief overview, we will note now and prove later that the geometric mean of two number is always less than the arithmetic mean and always more than the harmonic mean.

$$H(a,b) = \frac{2}{\frac{1}{a}+\frac{1}{b}} \leq \sqrt{a*b} \leq \frac{a+b}{2} = A(a,b)$$

Using this fact, we will approximate the square root of a number using the average of the arithmetic mean and the harmonic mean to provide an estimate for the geometric mean.

$$\sqrt{a*b} \approx \frac{\frac{2}{\frac{1}{a}+\frac{1}{b}}+\frac{a+b}{2}}{2} = \frac{H+A}{2}$$

While this is the essence of the method, this representation is not very pleasing. After algebraic manipulation, the form that we will use most often in this article will be:

$$\sqrt{a*b} \approx \frac{a^2+6ab+b^2}{4a+4b}$$

Note that usually one is more interested in taking the square root of a single number rather than two numbers. While one can pick a and b any arbitrary pair that multiply to the desired number, a common choice will be to set $b = 1$ as follows:

$$\sqrt{a} \approx \frac{a^2+6a+1}{4a+4}$$

If one performs a very precise answer after a single iteration, one should choose a and b to be very close to the actual root of the number in question. In the analysis that follows in the succeeding sessions, our standard initial guess will be using $b = 1$.

The formula above is not quite in the form of an iterative method, so we will express the complete method as follows:

1. Let $\sqrt{x}$ be the desired approximation
2. First iteration use:

$$x_0 = \frac{x^2+6x+1}{4x+4}$$

3. Following iterations:

$$x_n = x/x_{n-1}$$

$$x_{n+1} = \frac{x_n^2+6x_nx_{n-1}+x_{n-1}^2}{4x_n+4x_{n-1}}$$

As one proceeds through this iteration, we will show that $x_n \to \sqrt{x}$

II. Proof of Convergence

To prove that our result convergence to the desired root we will need to prove two simple inequalities. First, we will have to show that our two initial guesses straddle the square root of our particular number. Second, we must show that the arithmetic mean and harmonic mean

straddle the geometric mean. Our method relies upon that inequality to provide us with a particular approximation, our next approximation is found by dividing our original number by the approximation. The inequality for this straddling is fairly straight forward (choosing the less-than inequaity without loss of generality):

$$x_n \leq \sqrt{x}$$

$$\Rightarrow x_n^2 \leq x_n\sqrt{x} \leq x$$

$$\Rightarrow x_n \leq \sqrt{x} \leq \frac{x}{x_n}$$

The next thing that we need to prove for convergence is that the arithmetic and harmonic mean always straddle the geometric mean. In other words, we will need to prove:

$$\frac{2}{\frac{1}{a}+\frac{1}{b}} \leq \sqrt{a * b} \leq \frac{a+b}{2}$$

We will not attempt to prove the general case in this section, it's not necessarily required. These proofs are not novel, but we will carry them out anyways. We will first prove that the arithmetic mean is greater than the geometric mean. Suppose:

$$0 \leq (a-b)^2$$

Arithmetic on the RHS leads to:

$$\begin{aligned}(a-b)^2 &= a^2 - 2ab + b^2 \\ &= a^2 + 2ab + b^2 - 4ab \\ &= (a+b)^2 - 4ab\end{aligned}$$

Returning to the inequality, we get that

$$\begin{aligned}\Rightarrow 4ab &\leq (a+b)^2 \\ \Rightarrow \sqrt{ab} &\leq \frac{a+b}{2}\end{aligned}$$

Next, we will prove that the geometric mean is larger than the harmonic mean. This proof will rely on the results from the previous proof.

$$\text{Note that } H(a,b) = \frac{2}{\frac{1}{a}+\frac{1}{b}} = \frac{2ab}{a+b}$$

$$\begin{aligned}\sqrt{ab} &\leq \frac{a+b}{2} \\ \Rightarrow \frac{2\sqrt{ab}}{a+b} &\leq 1 \\ \Rightarrow \frac{2ab}{a+b} &\leq \sqrt{ab}\end{aligned}$$

This ends the proof for the required inequality. Together, these proofs mean that the approximations that we are using in our approximation straddle the square root, and that the method which acquires the next two guesses also straddle the square root.

III. Method Analysis

The program is easily enough integrated into Matlab or some other computing program, so we will not put an example code here. Instead, a table will be given to hopefully demonstrate the convergence of the method.

Table 1: *Example table*

n	I = 1	2	3	Actual
79	20.9875	9.3796	8.8882	8.88819
213	54.4953	18.2480	14.5945	14.59452

One can see from the table, the initial guess can sometimes be very far away from the actual answer. This is a result of that fact that we take our initial guesses to be 1 and n. Indeed, the choice of initial values, as we will see later, can have an impact on the number of iterations required for a certain precision. Another thing that one should notice from the table is that, as we will see later, the method converges really quickly once we get within a radius of 1 or two of the actual root. In this instance, our method was exact up to 4 decimal places in by the 3rd iteration. The 4th iteration produces accuracy of up to 14 decimal places. The next table demonstrates how many iterations are necessary to reach an accuracy of at least 10 decimal places.

Table 2: *Example table*

n	Iterations	Time (seconds)
10	3	3.41433×10^{-6}
100	4	4.3455×10^{-6}
1000	5	4.6558×10^{-6}
10000	6	5.5870×10^{-5}

Note that this does not mean that after 3 iterations our method is accurate to 10 decimal places when approximating the square root of 10, it means that we have at least 10 decimal places of precision. The actual error is 16 decimal places. That was just the first iteration at which we got better than 10 decimal places of precision. We will show later this can be a problem when studying the convergence rate.

Table 2 also suggests that there is some logarithmic relationship between n and the number of iterations. One also notices that each iteration doesnt necessarily add too much to the computation time. One should note that a graph of the time reveals a slightly erratic relationship.

To determine the convergence rate in a more analytical fashion, we will attempt to study it using Matlab. Specifically, we will define the convergence rate as being the smallest positive integer r such that

$$\frac{|e_{n+1}|}{|e_n|^{r+1}} = g^{(r+1)}(c)$$

In other words, we will attempt to find the first non-zero derivative of our function at the fixed point (the square root of the desired number. In order to do this, consider the following formulation of our method, with n being the number that we are approximating the root of.

$$f(a) = \frac{a^2+6a(n/a)+(n/a)^2}{4(a+n/a)}$$

The first, four derivatives are:

Table 3: *Example table*

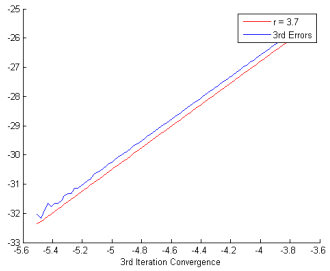
$f^{(r)}(a)$	$f^{(r)}(\sqrt{n})$
$\frac{(a^2-n)^3)}{4a^2(a^2+n)^2}$	0
$\frac{(a^2-n)^2)n(5a^2+n)}{2a^3(a^2+n)^3}$	0
$\frac{3n(5a^8-20a^6n+10a^4n^2+4a^2n^3+n^4)}{2a^4(a^2+n)^4}$	0
$\frac{6n(5a^{10}-35a^8n+30a^6n^2+10a^4n^3+5a^2n^4+n^5)}{a^5(a^2+n)^5}$	$\frac{3}{n^{3/2}}$

As the table illustrates, the first three derivatives vanish at the desired root for all numbers n. The fourth derivative is the first derivative where the evalution is a non-zero number. Therefore, we conclude analytically that the convergence rate is of the fourth order.

Analytically this can be seen easily, but computationally the result is much more difficult. There are several complications when trying to analyze the convergence numerically. First, in order to do so, we set up a log-plot of the errors. In short, in the following graph the equation of the line will look like

$$\log|e_{n+1}| = r * \log|e_n| + \beta$$

Where r is an approximation for the convergence rate, and β is some constant due to error in the data. Setting it up like this, we can create a linear regression of the errors and plot a best fit line that will give us an approximation for the convergence rate. This step can be tricky if there is not enough data, the data is noisy, or if the method converges too quickly. For example, if the method convergence too quickly and produces error of 0 in relation to machine precision, the log-plot will not function properly.
In addition, the convergence of this method is not linear. The error changes as the number of iterations increases. While we would like to see the result after many iterations, we are limited because of the aforementioned precision of the machine. Therefore, we only get a snapshot of the convergence rate.



This graph represents the approximation of the convergence rate for our method. Notice that this is the log-plot of the errors after three iterations. This was the best collection of points that we could get and have a fairly

consistent data set. One should then expect that this approximation will not necessarily be a good one. The result, however, matches with our expectations well. Analytically we expected a convergence rate of about 4, and we found numerically 3.7. We can then reasonably expect that this method is 4th order convergent.

IV. Comparison

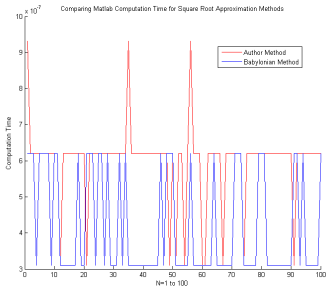
This section will serve as the crux of this paper. While already showing some examples of the convergence of the new method, it is not entirely useful until its compared to an already popular method. The Babylonian Method is defined as:

$$x_0 \approx \sqrt{n}$$
$$x_{k+1} = \frac{1}{2}(x_k + \frac{n}{x_k})$$

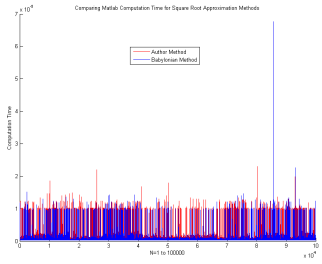
This also happens to be equivalent to Newtons Method for the particular function $f(x) = n - x^2$. This method was chosen because its well documented, and the convergence is known. This makes it easier to compare with our method. We will not discuss how Matlab or other programming software evaluate square roots because these are not iterative methods. In order to determine and weigh the methods, we will consider two different key criterion: number of iterations and time to reach a specific accuracy - usually 10^{15}. We will not specifically analyze the Babylonian Method by itself, only in comparison to the new method.

I. Time

One of the most important criterion for the success of an iterative method is the time it takes to complete a particular iteration. While one method may take less iterations than another, if the iterations are very computationally intensive, it may not be worth it. One will notice, if looking back at the formula for the new method, it is slightly more complicated than the Babylonian Method. Therefore, we will expect the computation time for that method to be slower than for the Babylonian. This, as the next graphs show, is not entirely clear. One should note that the computational time is very sporadic when we are dealing with such small scales.



While it appears that the Babylonian Method is usually quicker, the times only vary by about .3 micro seconds on average. Thats such a small gap that its hardly noticeable for most approximations. The preceding graph gives the times for a small window in order to show the micro-scale of the time differences. The next graph shows the times over a much larger set.

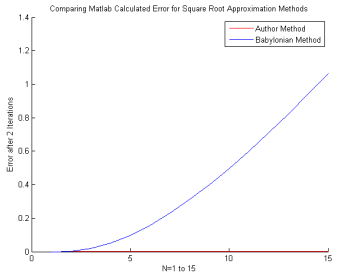


Aside from one abnormality, the two methods seem to be nearly indistinguishable at that level. While not conclusively better or worse, its still important that the computational complexity are roughly equivalent to the computer. This result is important because, as we will test in the next section, if one of them is more precise then it will not be because of trade-off with simplicity.

II. Error

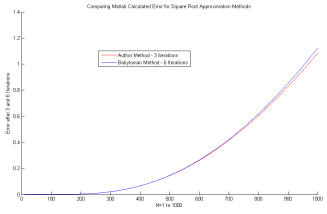
Arguably more important than the computational complexity is the error after each itera-

tion. When it comes to total time for a particular precision, having to calculate another iteration may cost more time than desired. If one of these methods can maintain lower error than it might need less iterations to reach a desired precision. In the long run, that could mean less computation time. In the next graphs, we compared the error of both methods after a certain number of iterations. We kept the initial values the same in order to have a good comparison.



This first graph shows the amount of error in calculation simple numbers after two iterations. The results are quite dramatic. The Babylonian method is by far more inaccurate for such a interval. It is so inacurate in comparison that the scale is hard to graph. The error of the new method appears to be near zero in comparison.

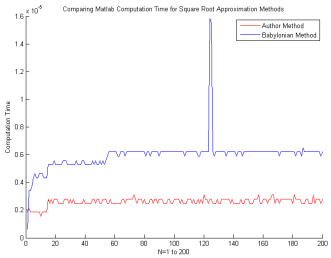
An interesting comparison is to show the difference between two iterations of the new method and 6 iterations of the Babylonian.



With this particular comparison, the two methods essentially become identical for a certain interval, but diverge for a sufficiently large n. This results implies that two iterations of the Babylonian Method produces the same accuracy as one iteration of the new method.

Since the Babylonian Method has a quadratic rate of convergence (since it is also Newton's Method), that leads us to believe that the new method has slightly better than double that rate, or quartic convergence. This matches well with the results found in the previous section.

It is then clear that the new method is superior to the Babylonian method in terms of error after each iteration. It takes half as many iterations to achieve a similar approximation. In order to bring together time and number of iterations we will compare the time it takes to reach a particular precision. In this example, the desired precision will be 10^{-15}. We expect that the new method will be much faster. Since the time per iterations is roughly equivalent, the fewer iterations will make the difference.



The graph shows what we thought would be true. While the times still do not vary by much - only 2 or 3 microseconds - there is a distinct difference between them. The erratic nature of the previous time graphs do not interfere with our interpretation, it is much clearer to see which method is slower.

Our closing remarks regarding the comparison to the two methods are as follows: The two methods have approximately the same computational complexity per iteration, the new method is far more accurate than the Babylonian Method per iteration, and the extra iterations that the Babylonian Method must do makes it slower overall for a desired precision. This does not particularly speak to the ability to do these approximations by hand, but by the processing of a computer.

These methods are not likely to be used on an actual computer, floating point representation is better. Therefore, the usefulness of this method is definitely obscure. Given that, an attempt to produce a more useful formula resulted in the method shown in the next section.

V. Nth Root Formula

As mentioned above, a iterative square root algorithm is probably too quaint to be useful in the era of modern computing power. In an attempt to generalize past square roots, we will produce a formula for the nth root of a number. The derivation is similar to that used above, relying on the inequalities of the Pythagorean means. In particular, we note that:

$$\frac{n}{\frac{1}{x_1}+\cdots+\frac{1}{x_n}} \leq \sqrt[n]{x_1 \cdots x_n} \leq \frac{x_1+\cdots+x_n}{n}$$

After doing some algebra involving the mean of the two saddling inequalities, we get the following formula:

$$\sqrt[n]{k} \approx \frac{xkn^2+x(n-1+k/x)(k(n-1)+x^n)}{2n(x^n+k(n-1))}$$

In this expression, treat x as a guess of the nth root of k. We will not go into the specifics of this formula, especially the convergence or limitations of this formula (the method wont work when n=0 for example).

While the nth root formula is more general than the square root method gave earlier, depending on the calculation complexity and the time required to perform each iteration it may be less useful.